TECHWIZ 5
A GLOBAL I.T. COMPETITION

Aptech®
Unleash your potential

AP GENIE
FINANCIAL ASSISTANT

Prepared by: T-Rex

Table of Contents

PROBLEM DEFINITION

The **ApGENie** chatbot is designed to facilitate seamless interaction between users and documents. It allows users to upload PDF and ask questions related to the content. The system extracts key information from the files, processes the data, and provides accurate, contextually-relevant responses.

This project addresses the challenge of interacting with unstructured data within documents, enabling users to query uploaded files effectively without manual reading or searching. **ApGENie** can be applied in financial, educational, research, legal, and corporate environments where extensive documents require analysis and quick information retrieval.

BACKGROUND

Financial services industry is continually evolving, driven by advancements in technology and increasing consumer expectations for faster, more accurate, and personalized services. Traditional methods of handling and processing loan related queries, such as those for housing loans, education loans, vehicle loans, and medical loans are often labor-intensive and time-consuming. These methods can lead to delays, inaccuracies, and a lack of personalized customer service, which in turn can result in customer dissatisfaction and operational inefficiencies.

In this context, the use of Generative AI (Gen AI) and Large Language Models (LLMs) such as those provided by OpenAI present a transformative opportunity. Gen AI and LLMs can automate and streamline the processing of complex and varied customer queries. By leveraging these advanced AI technologies, the proposed application can offer accurate responses to user queries, thereby enhancing the user experience and operational efficiency. The integration of Gen AI and LLMs ensures that the application can understand and respond to user queries providing comprehensive and relevant information tailored to each user's specific requirements. This innovation not only addresses the current limitations in handling large volumes of diverse loan-related queries, but also sets a new standard for customer service in the financial sector.

SCOPE OF PROJECT

The **‘ApGENie’** application should be able to assist users process PDF documents related to various loan types such as housing loans, education loans, vehicle loans, and medical loans.

Key features will include OCR and NLP for text extraction, data preprocessing, and conversion of text data into embeddings using pre-trained models. Advanced search techniques and an interface will be employed to provide accurate and relevant real-time responses to user queries, ensuring a seamless user experience.

FUNCTIONAL REQUIRMENTS

The project involves developing an intelligent question-and-answer application utilizing Gen AI, LLMs, and technologies from OpenAI. The application will process user input from PDF documents, enabling it to handle various structured and unstructured data formats commonly found in financial documents.

Some of the functional requirements are explained as follows:

- **PDF Input Handling** - Users will upload PDF documents containing .
- **Queries or information** - related to different loan categories. The application will employ OCR to extract text data from the PDFs provided.
- **Text Preprocessing** - The extracted text will undergo preprocessing to clean and structure the data making it suitable for analysis. This includes tasks such as tokenization, normalization, and removal of irrelevant content.
- **Chunking Strategies** - The preprocessed text will be segmented into meaningful chunks to facilitate better understanding and processing by the AI models.
- **Embedding Models** - The segmented text chunks will be converted into dense vector representations using advanced embedding models. These embeddings capture the semantic meaning of the text enabling effective comparison and retrieval.
- **Query Processing** - When a user submits a query, the application will process it using Gen AI and LLMs to understand the intent and context.
- **The query** will be matched against the relevant text chunks to find the most pertinent information.
- **Re-Ranking** - The retrieved text chunks will be re-ranked based on their relevance to the query, ensuring that the most accurate and useful information is presented to the user.

- **Response Generation** - Using Gen AI, the application will generate a coherent and contextually appropriate response to the user's query. This response will be tailored to the specific requirements of the user, providing comprehensive and precise information.
- **User Interface** - The response will be delivered to the user through an intuitive Web interface, allowing for easy interaction and follow-up queries if required.

The **'ApGENie'** will receive input in the form of PDF documents and users will provide queries. It will scan these documents and respond by extracting information directly from the PDFs.

NON-FUNCTIONAL REQUIRMENTS

There are several non-functional requirements that should be fulfilled by the application. These are listed as follows:

1. Scalable: The application must handle increasing numbers of users and documents without significant degradation in performance.

2. Secure: The application should be designed to prevent unauthorized users from gaining access to it.

3. Accuracy: OCR and NLP processes should achieve high accuracy rates in text extraction and understanding loan-related content.

4. Performance: The application should respond to user queries within seconds, even with large PDF documents.

5. Usable: The UI should be intuitive and accessible, facilitating easy interaction for users of varying technical backgrounds.

PROPOSED SOLUTION

‘ApGENie’ aims to develop an intelligent application that efficiently handles user queries about various loan types using GenAI, LLM, and OpenAI technologies. This application will accept PDF documents as inputs from categories such as housing loans, education loans, vehicle loans, and medical loans. The PDF files for these categories are provided for this project. The application will use Optical Character Recognition (OCR) and Natural Language Processing (NLP) to extract and digitize relevant text and data from PDFs for further processing. The extracted data will be preprocessed to clean and remove noise, irrelevant information, and inconsistencies ensuring high-quality and accurate data for subsequent stages.

Key features and variables relevant to each loan type will be extracted from the preprocessed data, forming the basis for understanding and processing user queries. The text data will be converted into embeddings using pre-trained models (OpenAI LLMs such as ChatGPT), providing a meaningful representation of the data and enabling the application to understand better and process the information.

User queries will be processed and matched with the extracted features from the data using chunking strategies, re-rankers, and generation prompts to effectively address the queries. An advanced search system, powered by Gen AI and LLMs, will leverage the processed data and embedding models to provide precise responses. An interface will be integrated to interact with users, using the processed data and model outputs to answer queries in real-time, ensuring a seamless user experience.

The final responses, search results, or generated text outputs will be presented to the users, ensuring they receive accurate and relevant information based on their queries.

DESIGN SPECIFICATION

- **Frontend**:
  - UI built using HTML5, Bootstrap 5, and JavaScript to offer an intuitive interface.
  - File upload section supporting PDFs.
  - User input form for submitting questions based on uploaded content.
  - Real-time loader animation for file processing.

- **Backend**:
  - Developed using Flask, Python.
  - Integrates Google Generative AI (Gemini) for language understanding and vector embeddings.
  - Uses FAISS for similarity search to provide contextually-relevant responses from document embeddings.
  - Langchain: A framework used to split documents, manage embeddings, and streamline AI query processing for efficient and context-aware responses.

- **Model**:
  - Google Gemini (gemini-1.5-pro-latest) for question answering.
  - Google Gen-AI Embeddings(embedding-001) for embeddings.

- **Database**:
  - Uses FAISS for vector storing in local system work as a database.

- **Key Features:**
    - PDF(s) file upload.
    - Ability to handle large documents by breaking text into chunks.
    - Conversational AI to answer questions based on document content.

- **QnA History Saving:**
    - Persistent Storage: Maintains a record of all user queries and AI responses.
    - Easy Retrieval: Allows users to revisit past questions and answers.
    - User-Friendly Interface: Presents the history in an organized manner for easy browsing.

DETAIL STEPS TO EXECUTE THE PROJECT

1. **Set up Environment:**
    - Install Python 3.10+.
    - Install Flask 3.0+.
    - Install google-generativeai.
    - Install langchain_google_genai.
    - Install langchain-community
    - Install few more necessary libraries: (mention in requirements.txt).

2. **Backend Setup:**
    - Clone the repository from GitHub.
    - Create a .env file and add the Google API key:

" GOOGLE_API_KEY=<your_google_api_key> "

3. **Run the Server:**
    - Execute python wsgi.py to start the backend.
4. **Frontend:**
    - Access the app on localhost:5000.
    - Upload files and ask questions.

5. **Deployment:**
   - Deploy using platforms like Heroku, AWS, PythonAnyWhere or any other server that supports it by following their deployment instructions for Flask applications.

TEST DATA USED IN THE PROJECT

PDF Files:

- Test using sample PDF files containing text such as financial reports, summaries, research papers, eBooks, and reports.

Project Installation Instructions

To Run Locally:

1. Clone the Repository:

bash Copy code

```bash
git clone https://github.com/username/project-repo.git
cd project-repo
```

2. **Install Dependencies:** Ensure Python and pip are installed, then run:

bash Copy code

```bash
pip install -r requirements.txt
```

3. Run the Flask Application:

bash Copy code

```bash
python server.py
```

4. Access the Web App:

   - Open your web browser and navigate to `http://localhost:5000`.

To Run on Google Colab:

1. Open the provided `.ipynb` file in Google Colab.
2. Install necessary libraries by running:

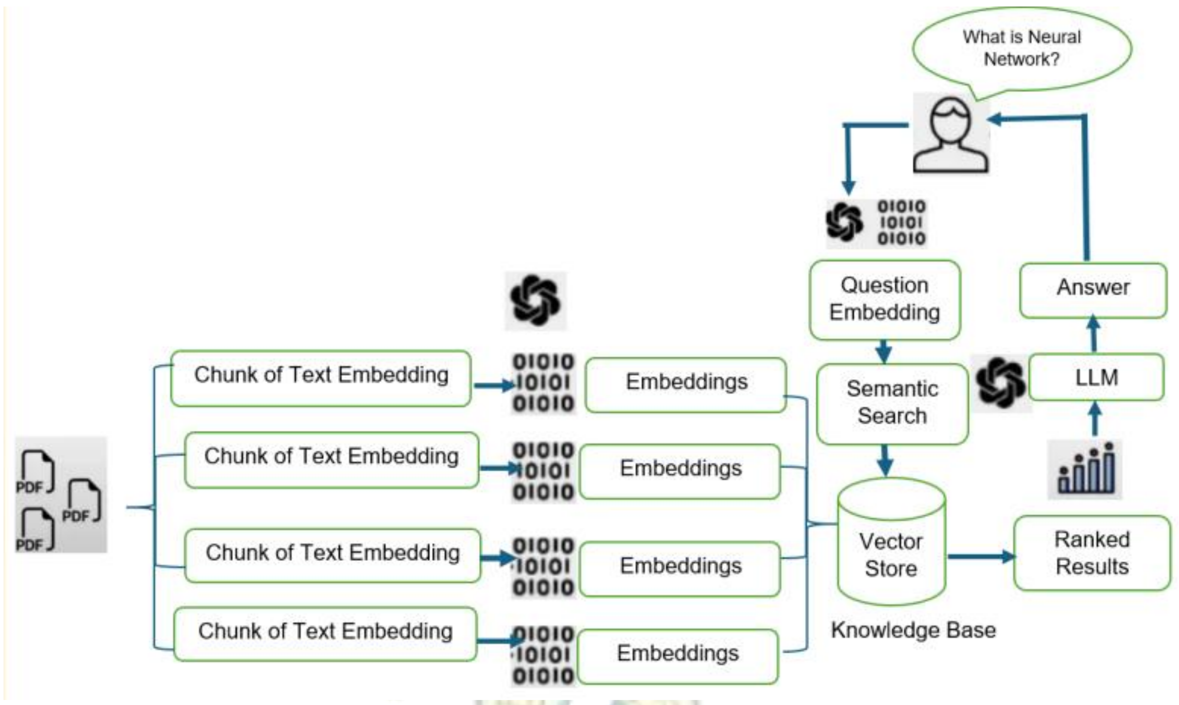
bash Copy code

```bash
!pip install flask google-generativeai
```

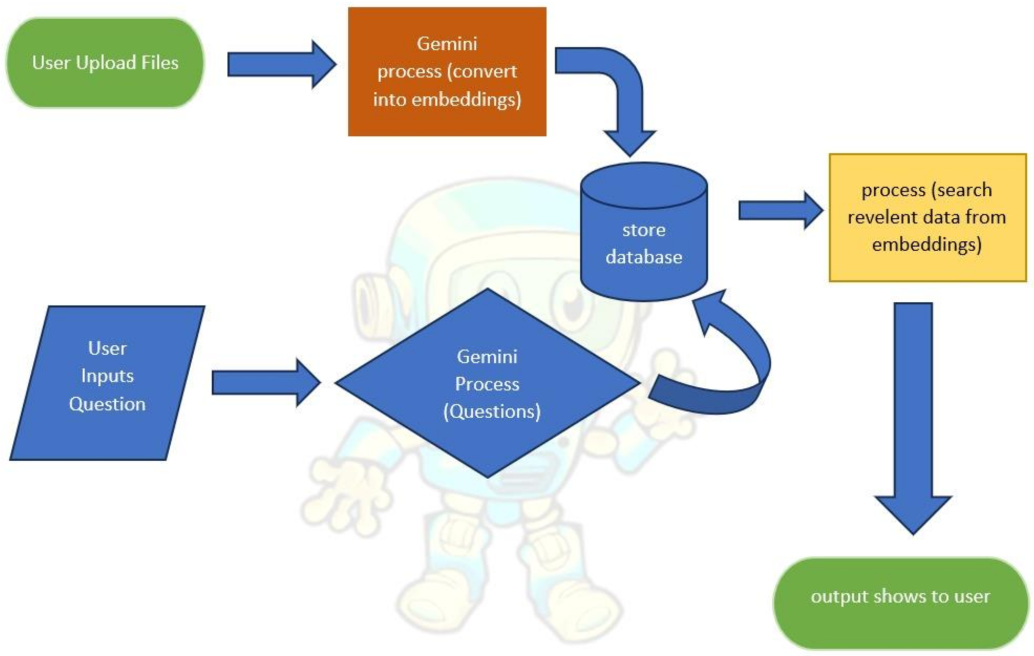
3. Follow the instructions in the notebook to interact with the API.

DIAGRAMS

Application Architecture:



Flow Chart:



User Upload Files
Gemini process (convert into embeddings)
store database
process (search revelent data from embeddings)
User Inputs Question
Gemini Process (Questions)
output shows to user

User Journey Map:

- User opens the chatbot interface.
- Uploads PDF Files.
- Receives a confirmation and a progress indicator while the file is being processed.
- Asks a question related to the uploaded content.
- ApGENie responds with a relevant answer based on document context.

GitHub Link:

You can access the complete source code on the GitHub repository:

https://github.com/ChouhdryRizwan/flaskAppGenei

Link to Published Blog:

The project is also documented in a detailed blog post on a free blogging platform. The blog covers the problem statement, LLMs, Gen-AI, Design choices and challenges faced during development.

https://apgenie7.wordpress.com/

Link to Deployed Application:

This is the link of deployed application.

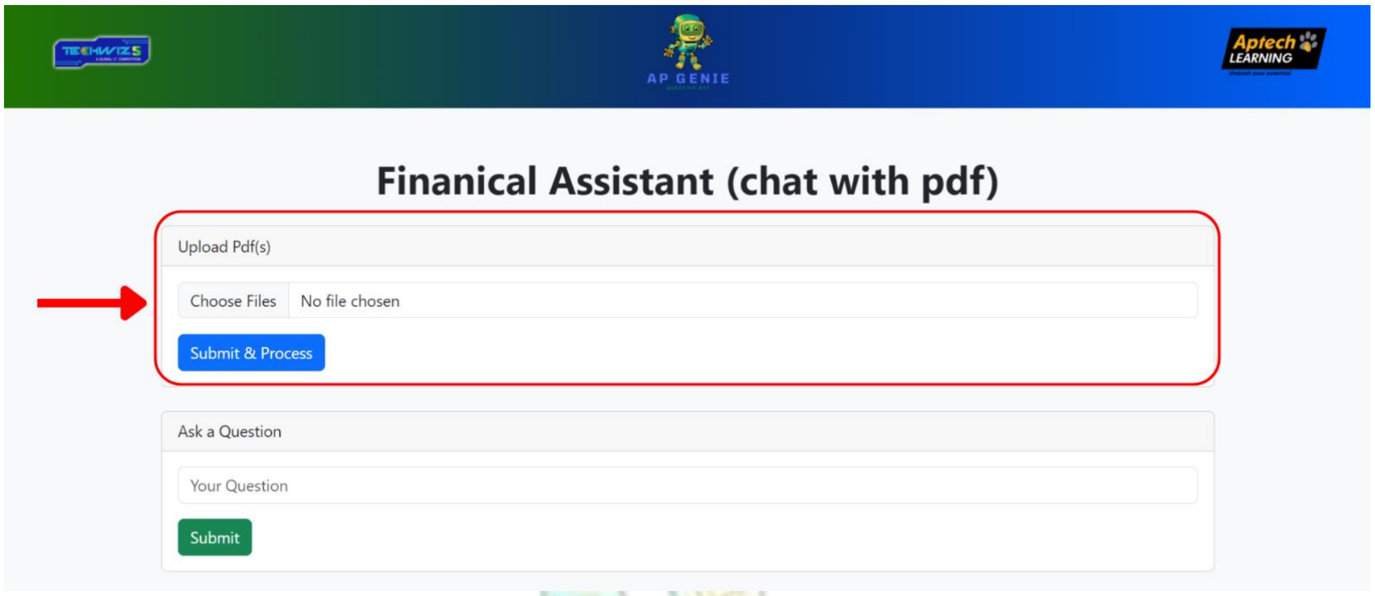
https://ap-genie.koyeb.app/

TASK ALLOCATION

Project Title	Project Ref. No.			Project Title
AP GENIE	S. No.	Name	Task	AP GENIE
	1	Muhammad Raheem	Langchain	
	2	Muhammad Taha	Gen-AI / Deployment	
	3	Anas Farooq	Flask / Testing	
	4	Saqib Noor	Data Stroage	
	5	Tanveer Haider	UI-Design	
	6	Khuraim Khan	Documentation/ GitHub	

SCREENSHOTS

Step 1: Open the application and upload PDF File(s).



Finanical Assistant (chat with pdf)

Upload Pdf(s)

Choose Files No file chosen

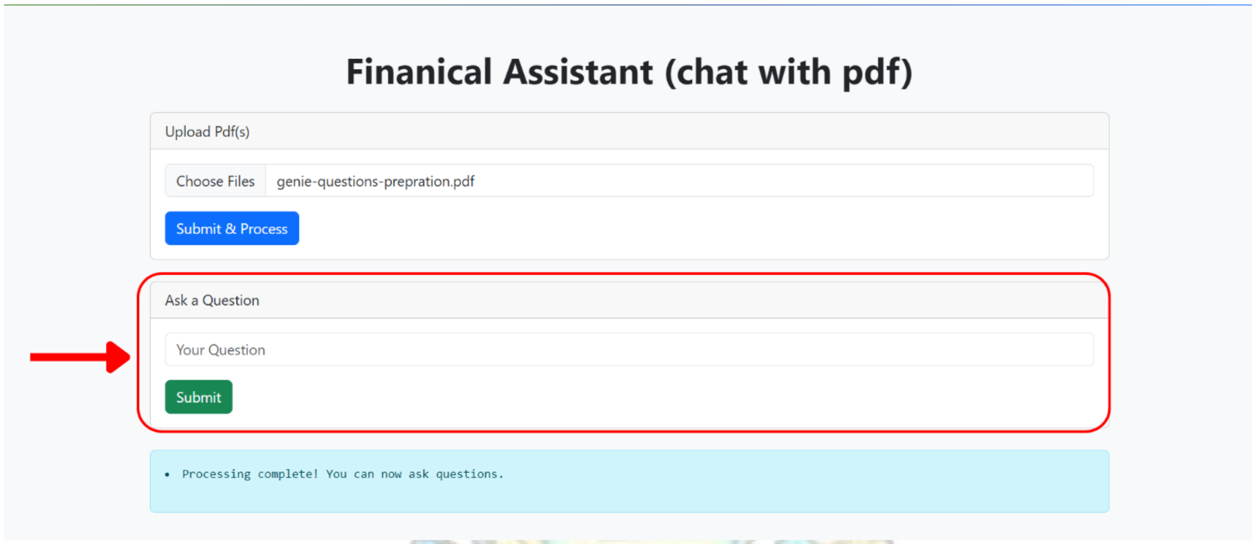
Submit & Process

Ask a Question

Your Question

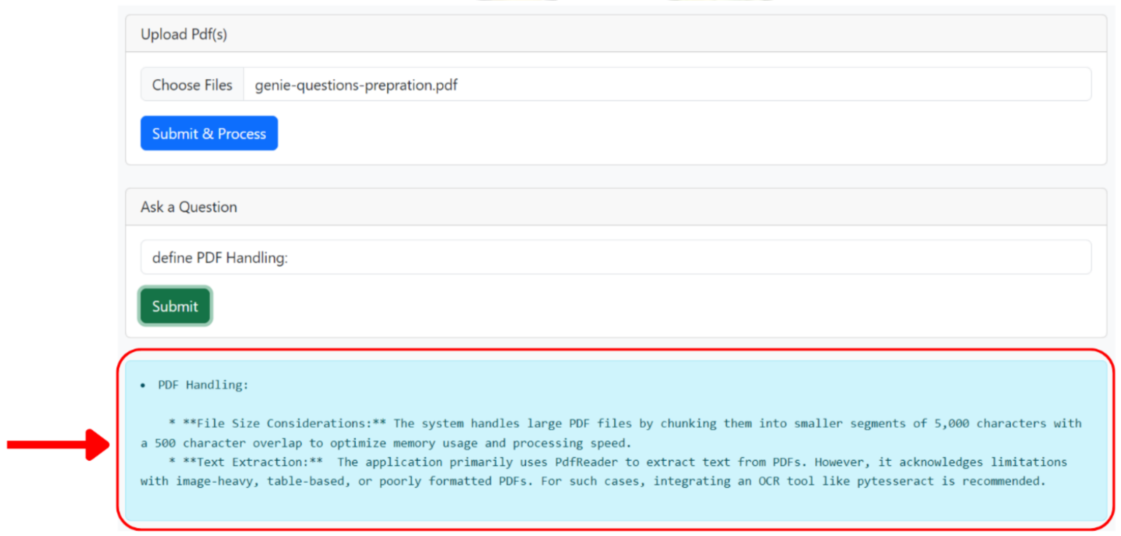
Submit

Step 2: Once file uploaded you can ask question.



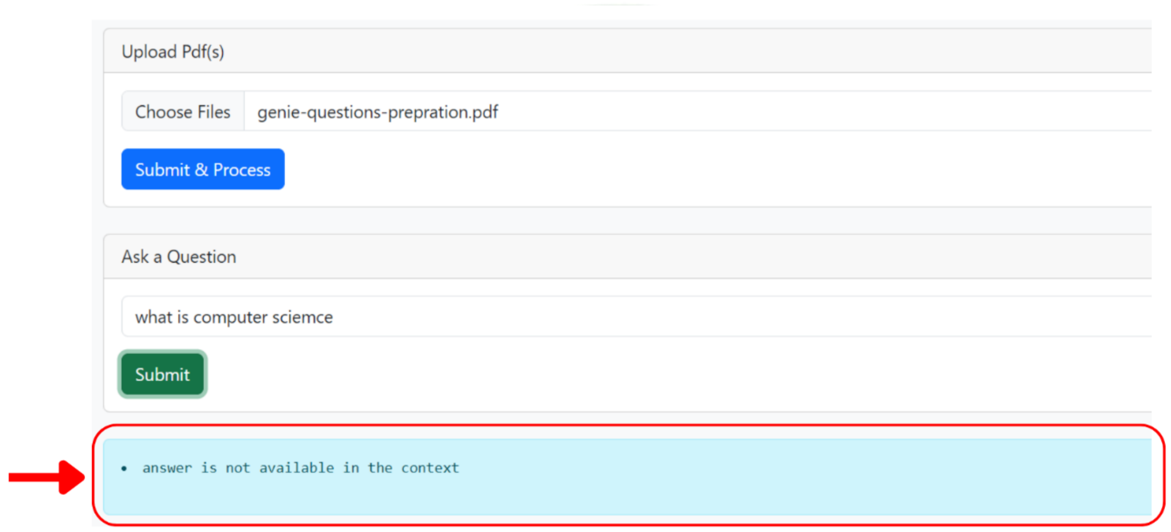
- Processing complete! You can now ask questions.

Step 3: If the answer is available in context, it will give you this response.



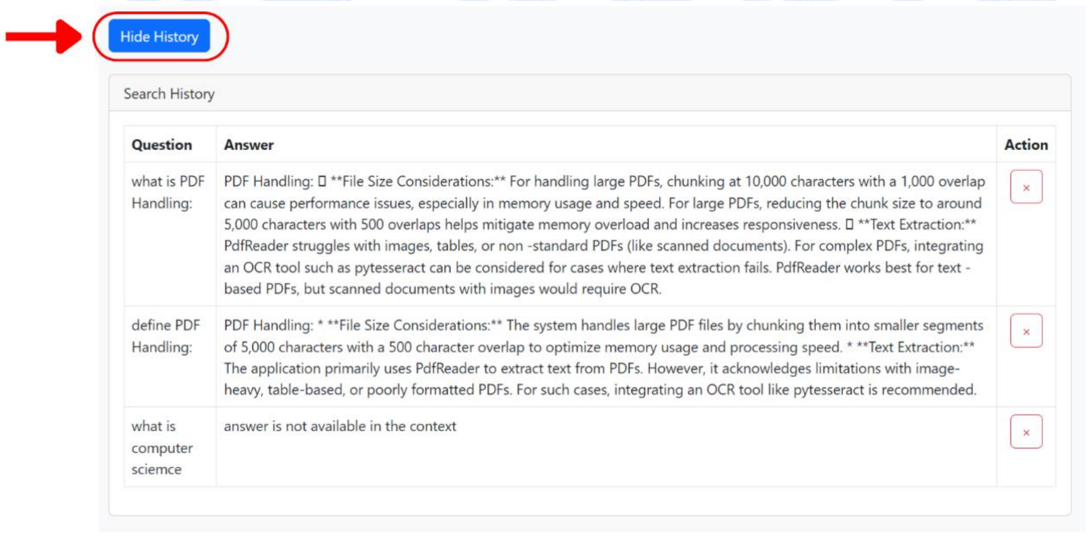
```
• PDF Handling:

    * **File Size Considerations:** The system handles large PDF files by chunking them into smaller segments of 5,000 characters with
a 500 character overlap to optimize memory usage and processing speed.
    * **Text Extraction:**  The application primarily uses PdfReader to extract text from PDFs. However, it acknowledges limitations
with image-heavy, table-based, or poorly formatted PDFs. For such cases, integrating an OCR tool like pytesseract is recommended.
```

Step 4: If your answer is not available in context, here is your response.



- answer is not available in the context

Step 5: You can view/manage chat history.



Question	Answer	Action
what is PDF Handling:	PDF Handling: ▯ **File Size Considerations:** For handling large PDFs, chunking at 10,000 characters with a 1,000 overlap can cause performance issues, especially in memory usage and speed. For large PDFs, reducing the chunk size to around 5,000 characters with 500 overlaps helps mitigate memory overload and increases responsiveness. ▯ **Text Extraction:** PdfReader struggles with images, tables, or non -standard PDFs (like scanned documents). For complex PDFs, integrating an OCR tool such as pytesseract can be considered for cases where text extraction fails. PdfReader works best for text - based PDFs, but scanned documents with images would require OCR.	×
define PDF Handling:	PDF Handling: * **File Size Considerations:** The system handles large PDF files by chunking them into smaller segments of 5,000 characters with a 500 character overlap to optimize memory usage and processing speed. * **Text Extraction:** The application primarily uses PdfReader to extract text from PDFs. However, it acknowledges limitations with image-heavy, table-based, or poorly formatted PDFs. For such cases, integrating an OCR tool like pytesseract is recommended.	×
what is computer sciemce	answer is not available in the context	×

PROJECT DELIVERABLES

You will design and build the project and submit it along with a complete project report that includes:

- Problem Definition
- Design Specifications
- Diagrams such as User Flow Diagram/User Journey Map
- Detailed steps to execute the project
- Test Data Used in the Project
- Project Installation Instructions
- Link of GitHub for accessing the uploaded project code (Link should have public access)
- Link of Published Blog

The source code, including .ipynb files for Jupyter Notebook and Google Colab, should be shared via GitHub. Appropriate access permissions should be granted to users to allow testing for Jupyter Notebook and Google Colab. The consolidated project must be submitted on GitHub with a ReadMe.doc file listing assumptions (if any) made at your end.

Provide the GitHub URL where the project has been uploaded for sharing. The repository on GitHub should have public access. Documentation is a very important part of the project; hence, all crucial aspects of the project must be documented properly. Ensure that documentation is complete and comprehensive. You should publish a blog of minimum 2000 words on any free blogging Website such as Blogger, Tumblr, Ghost or any other blogging Website. The link of the published blog should be submitted along with the project documentation.

HARDWARE REQUIREMENTS

Hardware:

Intel Core i5/i7 Processor or higher

8 GB RAM or higher

Color SVGA

500 GB Hard Disk space

Keyboard and Mouse

SOFTWARE REQUIREMENTS

Software:

Technologies to be used:

1. **Frontend:** HTML5 or any other scripting languages
2. **Backend:** Flask/Django
3. **Data Store**: PDF
4. **Programming/IDE:** Python, Jupyter Notebook, Anaconda, or Google Collab
5. **Libraries:** Tensorflow, NLTK, Keras, OpenAI API, Python libraries, and pre-trained transformers